Density Estimation of Nonequilibrium dynamics of the Ising Model Using Generative Neural Networks

Samuel D. Gelman and Guy Cohen*

School of Chemistry, Tel Aviv University, Tel Aviv 69978, Israel

(Dated: December 25, 2022)

Entropy estimation is a key struggle in the field of statistical mechanics. Systems of meaningful size require methods which can yield the entropy despite having a phase space which is grossly under-sampled. In this work a generative model, PixelCNN, was used to create a tractable distribution modeled after the Ising model in and out of equilibrium. The model proved to outperform existing methods both in accuracy and scaling. It's usefulness was demonstrated by exploring the role of entropy in non-equilibrium Glauber dynamics, describing the behavior of an Ising model within a rapidly oscillating magnetic field. The method seems capable of this level of performance for any system that can be well represented by pixelated data.

CONTENTS

Introduction:

I. BRIEF HISTORY OF ENTROPY

1. Thermodynamic formulation

Thermodynamics is the branch of science that investigates the states of matter. The foundations of thermodynamics were built on experimental observations and were used to formalize phenomena like heat, work and temperature. Initially explored by Sadi Carnot as a means to improve the efficiency of steam engines and later formalized by Lord Kelvin as the study of "thermo-dynamics" or the movement of heat between contiguous bodies [1].

Rudolf Clausius is known as one of the founders of thermodynamics and is attributed with the codification of the first two laws. The first law is the conservation of energy. It outlines a range of possible thermodynamic behaviors of a system. An isolated system, in a given state, with internal energy U_1, can only move to a new state with equal internal energy, such that $\Delta U = 0$. An understanding which relys just on this first law would allow for a multitude of reversible processes of the system to take place. The observed reality is far different. For any given set of parameters, for an isolated system, there exists only one well-defined equilibrium state and the system will evolve spontaneously and irreversibly to that state [2]. To explain this phenomena the second law of thermdynamics is needed.

Rudolf Clausius made the observation that heat cannot spontaneously pass from a colder to a hotter object without some other energy transfer happening at the same time [3]. This statement came to be recognized as one of the early statements reflecting the second law of thermodynamics. After over a decade of exploring the phenomena of interacting bodies and their transfer of heat, Rudolf Clausius coined the term entropy in 1865 to describe the energy lost as heat from any irreversible process [4].

Entropy was defined as being a state function which is additive for composite systems and is maximized at equilibrium. Thermodynamic relationships between entropy and the other thermodynamic state functions, namely internal energy and volume, were formulated throughout the 19th century by the likes of Herman Van Helmholtz and Josiah Willard Gibbs.

2. Statistical Mechanics and further abstraction

Ludwig Boltzmann drew a connecting thread between the macroscopic state functions, namely entropy, to the total number of microscopic states available to the system. He did this succinctly and beautifully and his equation for entropy,

$$S = k_B \log(\Omega), \tag{1}$$

appears on his tombstone in Vienna. Here Ω represents the total number of microstates available to the system and $k_B$is Boltzmann's constant, a constant which relates the average kinetic energy of a partic which reflects that it is the statistical amount le in a gas to the overall temperature of the gas. Boltzmann defines the degree of disorder and uses that to define the entropy of a system. This language is now common place when trying to describe the concept entropy.

Boltzmann anchored the thermodynamic definitions of energy and temperature to purely statistical observations of the system. But this statistical definition of entropy didn't only manifest itself in the discipline of thermodynamics.

Claude Shannon spawned an entire scientific theory when he published his paper in 1948 titled "A Mathematical Theory of Communication." Here Shannon tries to quantify the amount of information contained in a given observation and layout how that effects its ability to be communicated. He defines his conception of informational entropy with a formula extremely similar to that of Boltzmann's famous theorem:

$$S = -K \sum_{i=1}^{k} p(i) \log(p(i)), \tag{2}$$

K is a positive constant which can be used as a normalization term, and p is the probability of event i happening. S is replacing the variable H used in the original paper to maintain consistency through this work.

The abstraction and applicability of the idea of entropy caused it to spread through multiple disciplines which is reflected by an increased adoption of the word after Shannons formulation as seen in Figure 1. Entropy entered into the minds of economist, sociologist, statisticians and computer scientists [6] [7] [8] [9].

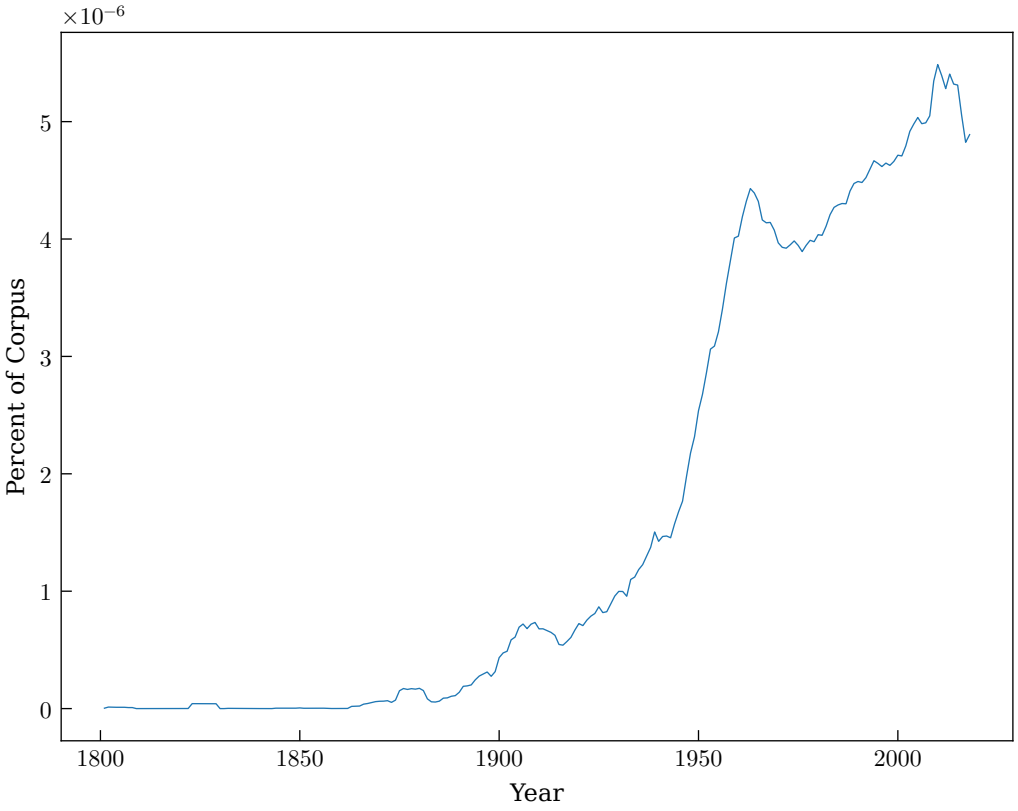

Figure 1. This plot shows the use of the word entropy over the last 218 years. A jump of 2.35x is seen after Shannon's landmark publication in 1948. The data is taken from the Googles Books Ngram project. The project was the result of cataloging the use of words and phrases from a vast corpus of digitized literature ranging several hundreds of years [5].

II. ENTROPY ESTIMATION

A. The Difficulties

When trying to learn about a system through experiments or simulations it is likely that the amount of observational data obtained will only make up a small portion of the total available sample space present. Often times it wont be known just how big that available space is. In other terms, samples $\{X_i\}$ can be obtained from a distribution $P(X) = \frac{p(X)}{Z}$, where that distribution is normalized by some value Z, referred to as the partition function, which is not necessarily known. Without knowing Z, $p(X)$ gives the unnormalized

distribution of $P(X)$. Through this limited vantage into the configuration space, some values can be estimated, for example the internal energy U:

$$U = \int \mathrm{d}X\, P(X)\, E(X) = \frac{1}{N} \sum_i E(X_i) + O\left(\frac{1}{\sqrt{N}}\right). \tag{3}$$

Where X is the random variable which contains all possible configuration of the system and $E(X)$ is the energy of the system at that state. X_i is the i_{th} sample of the N total samples taken. These expectation values approach the true value with an error of $O\left(\frac{1}{\sqrt{N}}\right)$ so relatively good estimations can be reached independent of the size of the system [10]. There are many uses to defining these properties [11] but there are limits to what one can learn from them about the system as a whole. For certain system properties the full distribution must be known.

Calculating entropy, S, requires $P(X)$ to be known explicitly:

$$S = \int \mathrm{d}X\, P(X) \ln(P(X)) = \frac{1}{N} \sum_i \ln(P(X_i)) + O\left(\frac{1}{\sqrt{N}}\right). \tag{4}$$

Analytically calculating the entropy for systems of meaningful sizes grows intractable quickly [12], solving for the partition function Z simply becomes too difficult. In order to calculate the analytical entropy, the probability density function must be known and an integral must be taken over all possible states.

B. Past work

There is a long standing tradition of attempting to overcome this barrier using computational brute force through methods like molecular dynamics simulations [11] or Monte Carlo sampling [13]. These approaches become ineffective when the space of exploration reaches a meaningful size because of how the configuration space scales with system size [14].

This led many algorithms to limit themselves to particular regimes in equilibrium [12]. In thermal equilibrium, energy fully determines the probability of occupying a configuration and several powerful, generic techniques have been developed to take advantage of this [15].

The broad histogram approach leverages this connection by taking random walks through energy space while constructing a distribution which approaches that of the density of states [16]. This method has been built upon to make substantial progress in the field [17] [18] [19] [20] [21]. Yet the method is restricted to distributions in equilibrium.

Estimating the entropy by making small changes to the analytic solution has also seen success [22, 23] [24] [25]. Work has been done to extend this method to non-equilibrium states but remains restricted in reach [26]. However, for non-equilibrium systems, even when simulation is possible, the probability density generally cannot be expressed in terms of a single, known variable [27], [28].

Most approaches therefore resort to enumerating the configuration space [29], for example by coincidence counting. This method quickly runs into issues for large systems [30].

Another more recent method measured the degree of compressibility to measure the entropy [31] by using lossless compression algorithms like zip and Lempel-Ziv compression.

III. NEURAL NETWORKS

A. Brief History

The term machine learning first entered scientific publication in 1959 by Arthur Samuel [32]. Samuel's work came in the wake of research funded by the US Navy where Frank Rosenblatt developed the perceptron [33]. The first "neural networks" (NN) were built out of a collection of these perceptrons, or binary classifiers that define a threshold function where the input is some real-valued vector and the output is a binary value:

$$f(\mathbf{x}) = \begin{cases} 1 & if : \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & otherwise \end{cases}, \tag{5}$$

where $\mathbf{x}$ is a real-valued vector of length i and $\mathbf{w}$ is a vector of the same size with tunable weights. There are several methods for tuning the weights during training but the most popular is an algorithm named gradient descent which is often attributed to Augustin-Louis Cauchy. The concept of the perceptron was an attempt to mimic the workings of the mammalian brain with the aspirations to create a generalized artificial intelligence.

Soon after these findings, computers were being put to work on a wide range of difficult tasks such as signal detection, pattern recognition and error detection [34] [35] [36].

The field declined in popularity as the dreams of NN were unrealized due to hardware and algorithmic constrictions. Metal-oxide-semiconductors along with other technologies enabled a spike in affordable and available processing power for the training of NN in the

1980s [37]. Paul John Werbos described a process of training artificial NNs in which each training example attributes to a change in the weights relative proportionally to what would cause the greatest decrease in the cost. The algorithm for calculating the gradients was called back propagation and it used the chain rule of derivatives which allowed it to work recursively through the depth of a network [38]. For this reason it became extremely important for the training of deep multi-layer NNs as it enabled for efficient tuning of the weights and biases back through the whole depth of the network.

Back-propagation coupled with a stochastic approximation method first investigated by Herbert Robbins and Sutton Monro allowed for gradient descent algorithms to converge to a minimum of the cost function within reasonable time frames [39].

Over the last twenty years the field has exploded with new algorithms and system architectures appearing out of both the academic and commercial domains. Models are excelling at image generation, natural language processing and computer vision. It only follows that they should be used as a tool for scientific inquiry.

Models

IV. CONTINUOUS PROBABILITY DISTRIBUTIONS

A. Distributions and density functions

When considering a system it is useful to chose a limited number of parameters to describe it. By using these parameters, a function can be constructed which approximates the system given a manageable amount of input. As a simple example one can parameterize the likelihood of it raining tomorrow based solely on the observation of whether or not it is raining today,

$$f_{\text{rain}}(x), x \in \{0, 1\}, \tag{6}$$

where 0 is not raining and 1 is raining. The binary input of whether it is currently raining is a means of simplifying the sample space into a discrete set of numbers, in the context of probability distributions this is known as a random variable, though the restrictions could be much less broad for, example $\mathbb{R}$ or $\mathbb{N}$.

To add a slight level of complexity, suppose another random variable is added which is the current percentage of cloud coverage,

$$f_{\text{rain}}(x, y), x \in \{0, 1\}, y \in [0, 1]. \tag{7}$$

Adding an additional parameter would likely add to the accuracy of the model being constructed. It would also increasing the amount work the observer would need to make in order supply input data for to the model.

Further axioms must be satisfied for these models to be characterized as valid probability distributions. These are known as Kolmogorov axioms [40]. When these axioms are met the function can be called a probability density function (PDF).

B. Distributions of increasing complexity

1. *Normal Distributions as Reference*

The Normal or Gaussian distribution is a special distribution in that it describes the normalized statistical behavior of many independent and identically distributed random variables even when those variables themselves are not normally distributed [41].

Normally distributed distributions can be described as:

$$\mathrm{N}_{\mu\sigma}(x) = Ae^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{8}$$

where μ is a well defined mean, σ^2 is the variance and $A = \frac{1}{\sigma\sqrt{2\pi}}$. The normal distribution is one which has a unique relationship relating the mean and standard deviation when taking the expectation of the distribution, namely:

$$\left\langle (x-\mu)^2 \right\rangle = \sigma^2. \tag{9}$$

This relationship makes the entropy of normal distribution extremely simple to calculate analytically. The differential entropy can be written in terms of the expectation as:

$$S(x) = \int_x f(x) \log f(x)\, dx = \langle [-\log(f(x))] \rangle. \tag{10}$$

[42] Utilizing this relationship, the analytical solution for the differential entropy can be derived as follows:

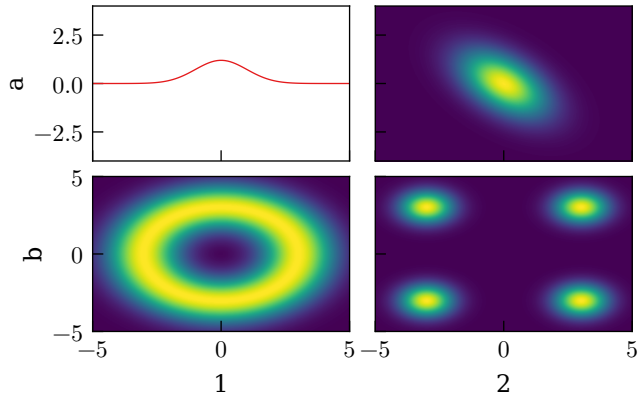

Figure 2. Shows model example of the continuous space functions used to exemplify the applicability and limitations of various density estimation and dimensional reduction techniques. panel (a1) is a simple, one dimensional normal distribution with mean 0 and standard deviation 1. panel (a2) is a two dimensional normal distribution where the two random variables are linearly correlated. panel (b1) is an example of a distribution with radial symmetry. panel (b2) is a distribution which has wells separated from one an other and symmetrically distributed in space.

$$S(x) = \langle [-\log(\mathrm{N}_{\mu\sigma}(x))] \rangle ,$$

$$= -\left\langle \left[\log\left(\left(2\pi\sigma^2\right)^{-\frac{1}{2}} e^{\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)} \right) \right] \right\rangle ,$$

$$= \frac{1}{2}\log\left(2\pi\sigma^2\right) + \frac{1}{2\sigma^2}\left\langle \left[(x-\mu)^2\right] \right\rangle ,$$

$$= \frac{1}{2}\left(\log\left(2\pi\sigma^2\right) + 1\right). \tag{11}$$

Using this analytical benchmark will allow for a standard with which to test various methods.

2. One Dimensional Normal Distribution

The one dimensional normal distribution, figure 2 panel (a1), is a simple example of a distribution which can be fully approximated using brute force methods like histogram binning. The fact that the analytic solution is known and easily calculated makes it an intuitive example to show that there are scenarios where simplistic methods can be effective.

3. Rotated Two Dimensional Gaussian

The rotated, two dimensional normal distribution, figure 2 panel (a2), is convenient because it is known that there is a transformation which can be done to remove the linear correlation between the two dimensions of the distribution. An attempt to reduce the problem to a product of two Gaussian distributions will fail using the presented basis vectors but will be successful after the proper transformation.

4. Radially Symmetric Distributions

The radially symmetric distribution. , figure 2 panel (b1), is similar to the rotated Gaussian in that dimensionality reduction is not a trivial task in one coordinate system while it is in another, in this case within a radial coordinate system. The algorithm which works on the rotated Gaussian, which finds the linear correlation of the data, will fail on this example.

5. Distribution of Symmetric Wells

The final toy model used, figure 2 panel (b2), is a case where the distribution is lacking a clear direction of correlation because of its symmetry. While having a radial symmetry is also has a certain degree of linear correlation. Because of this, it can dangerously provide false positives using the methods applied to the rotated Gaussian. For this reason it is used to show the effectiveness of other density estimation techniques.

C. Informational Entropy

From the lens of information theory, the amount of information can be described by the amount of surprise an observation gives the observer, weighted by the value of the observation. The entropy could then be described as the average amount of information gained per observation.

Observations with low likelihood give a lot of information but occur less often and are therefore weighted less. This is a function of the logarithmic nature of the calculation of

entropy in equation 2. Systems have their highest entropy when they are uniform or when all possibilities have an equal likelihood of taking place [22].

In order to calculate the entropy of a real world system the full PDF must be known which is often difficult or impossible even when sampling from that system might be as easy as checking the weather outside. Thankfully there are also ways to sample PDFs proportionally and with minimal computational force as well, allowing inexpensive and effective means towards testing algorithms and model efficiency.

D. Sampling: The Metropolis Hastings Monte Carlo Algorithm

As mentioned above, given a distribution $P(X) = \frac{p(X)}{Z}$, without knowing Z, samples, $\{X_i\}$, can still be drawn which accurately reflect the distribution. Experiments will almost always have this effect. Extending this ability to simulations was of optimal importance to increase the usefulness of computers as a tool for scientific investigation by properly leveraging the strength which they could have as tools for investigating phenomena and building accurate models.

In 1953 Nicholas Metropolis developed an algorithm that did just this for symmetrical distributions [43]. W.K. Hastings extended it to more general cases in 1970 [44].

The algorithm works by taking a random walk along a distribution where candidate step length and direction are randomly chosen based off predetermined criteria. The probability of executing the candidate step is more likely when moving from lower values of $p(X)$ to higher ones.

More specifically the algorithm starts with an initialization sample X. Then it purposes a new sample, X', which had been modified as described above. A uniform random number, $a \in [0, 1]$ is generated and compared to an acceptance function which also outputs a single real number between 0 and 1:

$$Q(X, X') = \min\left\{\frac{p(X')}{p(X)}, 1\right\}. \tag{12}$$

If $Q(X, X') \geq a$ then X' is accepted and the algorithm starts over, comparing a new candidate step to the previously accepted X'. In the event that a is larger than the output of the acceptance function, the algorithm resets and a new candidate step is measured against our original X. All of the accepted samples are stored an make up the data set proportional t

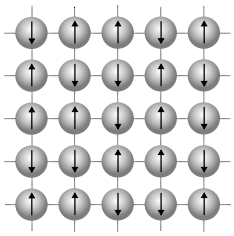

Figure 3. Shows a spin representation of an Ising model, where cells in the grid assume a binary value and are arranged in a spatially significant lattice structure [46].

the true population. The method can extend to many dimension though at high dimensions modifications are required to insure accurate sampling [45].

V. THE ISING MODEL: EQUILIBRIUM

A. Ising introduction

The two dimensional Ising model is constructed of sites which can take on one of two values. When the model sites takes on values of $\alpha \in [-1, 1]$ the model is often referred to as a spin glass model as the positive and negative values of a site point to an up or down spin respectively. When $\alpha \in [0, 1]$ the model is often referred to as a lattice gas model where α is the occupation number of a site, a partial can be said to occupy a site when $\alpha = 1$. This work will concern itself with the spin-glass perspective of the Ising model.

he model can be constructed in any number of dimensions greater than zero but is very often, as well as in the case of this work, used in two dimensions. Sites are arranged in a grid which is usually a square lattice. An illustration can be found in figure Figure 3 on page 14.

Any given configuration has a well defined energy as described by its Hamiltonian:

$$E(\alpha) = -\sum_{(i,j)} J_{i,j}\alpha_i\alpha_j - H\sum_j h_j\alpha_j. \tag{13}$$

The model described in equation 13 is a two dimensional lattice structure of an Ising model. The Hamiltonian, E, takes as an input α, which here is a spin configuration which has an index leading to each site of the system. J is known as the coupling constant and represents the reaction strength between neighboring spins. When $J > 0$, which it will be taken to

be for the scope of this work, and neighboring spins are of opposite values, the interaction energy will be positive, contributing energy to the system. The opposite is true for agreeing spins, meaning the model energetically favors uniformity of spins across the model. The second term of the Hamiltonian acts on sites individually and represents an external bias to the model. The magnitude of the bias is represented by H and can be thought of as an external magnetic field. The degree at which a given site is effected by the bias is represented by h. The susceptibility of a site, h, can be thought of as a magnetic dipole and will be held constant for all site for the scope of this work.

We use the Boltzmann distribution to find the configuration probability as described by equation (14):

$$P_\beta(\alpha) = \frac{e^{-\beta E(\alpha)}}{Z_\beta}, \tag{14}$$

where β is the inverse temperature, $\beta = \frac{1}{k_B T}$ and k_B is Boltzmann's constant. The system is always tending to lower energy. Heat disrupts the tendency towards a total agreement of spins. In other words, the higher the temperature, the lower the correlations between the spin orientations of sites.

There exists a temperature where the a fundamental change happens within the model, where sites a less likely to be correlated then not. To use the conceptualization of the Ising model representing magnetic spins, it is the point where the average mean magnetization of the system goes to zero. This point not simply a construct within theoretical models. The phenomena was first observed and recorded by Pierre Curie just before the turn of the 19th century [47]. He observed that the ferromagnetic properties of iron we lost at a high temperatures and proceeded to measure a catalog this temperature for many metals. This point is known as the Curie temperature and can be seen in figure 4.

The temperature marks a point where the system moves from an ordered to a disordered state. The shift from ordered to disordered can be classified as a phase transition and the point, moving through a specific dimension of parameter space, where a phase transition takes place is known as a critical point.

The presence of a Curie temperature in the Ising model as well as in solid metals begins to demonstrate the usefulness of the model and starts to explain its wide ranging use an popularity. This story is more beautifully closed because of the of a finding made a a little

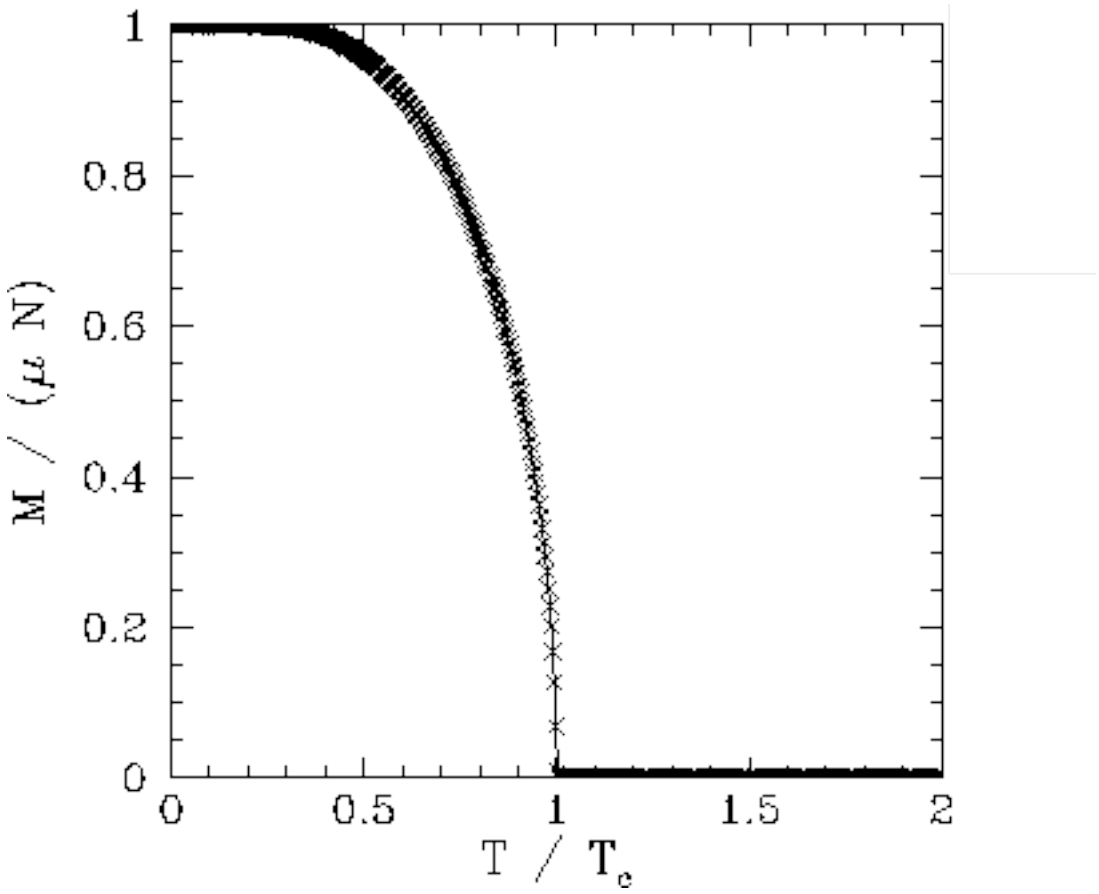

Figure 4. Shows the mean magnetization of the square Ising model as a function of the temperature. The temperature here is normalized by the curie temperature, T_c, such that 1 on the y-axis marks the critical point of phase transition. [48]

over two decades after the Ising model was introduced to the scientific community: an analytic solution to the models state functions.

B. Onsager's analytic solution at the limit of an infinite sized lattice

Lars Onsager, in 1944, solved for an analytic description of the 2d Ising model [49]. Using his formulation, the entropy is known as a function of the temperature and is available as an analytic benchmark for higher dimensional entropy estimation problems.

The analytic solution for the entropy of a 2d-lattice in the case where there is no external magnetic field in the thermodynamic limit is possible analytically calculable thanks to Onsager's solution. He laid out integrals to solve for the free energy and the internal energy

as follows:

$$-\beta F = ln(2) + \frac{1}{8\pi^2}\int_0^{2\pi} \mathrm{d}\theta_1 \int_0^{2\pi} \mathrm{d}\theta_2 \ln[\cosh(2\beta J)^2 - sinh(2\beta J)(cos(\theta_1) + cos(\theta_2))], \quad (15)$$

$$k = \sqrt{1 - \frac{1}{\sinh(2\beta)^2}}$$

,

$$U = \frac{-J}{\tanh(2\beta J)} \cdot \left(1 + \frac{2}{\pi}\left(2\tanh(2\beta J)^2 - 1\right) \cdot \int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1 - \frac{4k}{(1+k)^2 \sin(\theta)^2}}}\right). \quad (16)$$

From the free energy, F, and the internal energy, U, the entropy can be found by manipulating the expression for the Helmholtz free energy:

$$S = \beta(U - F)$$

[2]. This formulation is used to describe the isotropic case, where horizontal and vertical nearest neighbor interactions are equal. This formulation is at the thermodynamic limit, which means it reflects the entropy of an infinite sized lattice. Although they are similar, the system has different properties at finite size [50]. Correcting for the finite size limit is necessary to properly assess the traction and effectiveness of any approach working with sample data which is finite in size.

C. Analytic corrections for finite sized lattice

Methods have been developed to correct for the finite size effects on the entropy [51]. These approximation are used as the analytic benchmarks for the results and figures which follow in this work.

D. Insight from the Ising model

The two dimensional Ising model is often used as the starting point for investigation into entropy estimation techniques because of its relative simplicity, it's high dimensionality and the availability of an analytic solution.

Cooperative phenomena describe a group of phenomena caused by the interactions of many elementary particles [52]. The undertaking to model these systems is similar to the motivations which drove the statistical mechanical approach. The Ising model was the first and most effective example of these exactly solvable models. Is has been studied in depth and was famously observed to mimic the behavior of ferromagnetic materials, cooperative systems composed of a collection of atoms with magnetic dipoles [53]. Thorough investigations in to the characteristics of magnetic susceptibility have been of interest have yielded many conclusive results [54] [55] [56].

In addition to steady state estimations the time dependent statistics of the model have been studied at depth. These mechanics are explored in greater depth later in this work. The dynamics can lend insight to phase transition and models of diffusion dynamics [57][58].

These correlated systems and dynamics require the high dimension of models like the Ising model to be studied and are not obtainable from the low dimension continuous space functions mentioned above.

E. Ising Sample Generation

The well defined Hamiltonian makes sampling of the Ising model extremely simple because the partition function does not need to be known in order to compare the relative probability of two spin configurations. This can be done using popular algorithms such as the Metropolis-Hastings algorithm which is described in further detail in the Methods section.

The relative probability of finding the Ising model in one state as opposed to another is clearly outlined by the models well defined Hamiltonian, equation (13).

A reliable set of samples can be generated by taking snapshots of a time evolution of the system Metropolis *et al.* [43]. Here, "reliable" means a relatively small but representative fraction of the total population. In order to insure that samples are adequately uncorrelated, a sufficiently large amount of Monte Carlo steps should be waited between saving samples. These nearly uncorrelated samples represent the actual distribution and can be obtained with modest demands on computer time and power. For this work, the correlation of the samples were tested to ensure adequate waiting times and is described below in more detail.

For samples at high temperatures the Metropolis-Hastings algorithm was used. For low

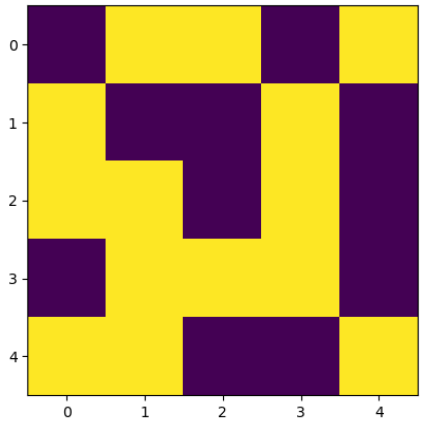

Figure 5. An example of a generated Ising sample with side length $L = 5$ where yellow represents one of the spin states and the purple represents the opposite one.

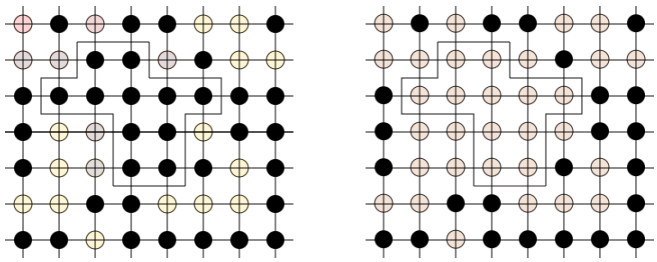

Figure 6. This is a representation of an example cluster using the Wolff algorithm. These clusters are easily defined for low temperatures and allows for sampling similarly likely configurations even if they as spatially distant from one another.

temperature samples, where changes take place more slowly, the Wolff algorithm produced low correlated samples much more reliably and in shorter computation time [59]. The algorithm works by leveraging the long distance correlations which become increasingly more dominant as low temperatures when the distribution is more strongly influenced by the relative energy of the configurations. This tendency causes algorithms like Metropolis-Hastings to get caught in local minima.

The Wolff Algorithm also works by stochastically exploring configuration space. Unlike the Metropolis-Hastings algorithm which explores possible configurations by making incremental changes to a single random variable at a time, the Wolff algorithm creates clusters and flips those entire clusters with a probability of 1.

The algorithm checks all possible links of a site to test whether or not the cluster will extend over that link. Links are accepted to become part of the enlarging cluster with a probability of

$$p_+ = 1 - e^{-2\beta\sigma_-\sigma_+}, \tag{17}$$

where $\sigma_-$and σ_+ are the spin values at either end of a given link. When spins are opposite, it is impossible for the link to be accepted in the cluster. For high temperatures, when β is low, the likelihood of forming large clusters also drops significantly.

When a cluster is being formed, each available link is only tired once. That said, a site can be included into the cluster via any of its links so even if it fails once it might still become included via the link of a different neighboring site.

This algorithm allows for a freer movement though the configuration space when long lasting correlations are present. That said, it is a more computationally expensive algorithm to run so it is only used for sample generation when necessary.

F. Correlation

Metropolis Monte-Carlo sample generation works by making incremental changes to the current configuration, comparing the relative energies, and then probabilistically deciding to accept or reject a sample using those energies as a metric. When building the data sets for this paper it was important that the samples chosen to be accepted from the Monte Carlo algorithm were uncorrelated. Given the incremental nature of the sampling algorithm, samples can easily be correlated one to the next if a sufficient number of Monte Carlo steps is not waited between accepting a sample into the data set. It is therefore important to have a metric for testing the level of correlation found between samples.

Samples, X, can be generated concurrently using several different computational cores where each core is running a separate realization of the Metropolis-Hastings or Wolff algorithm. The data can then be organized using three indices,

$$X_d^{n,i}, \tag{18}$$

where D is the total number of realizations, N is the total number of samples in each realization and I is the total number of pixels in each sample.

Given this structure, the mean value for an individual pixel can be calculated across realizations as:

$$\left\langle X^{n,i} \right\rangle = \frac{1}{D} \sum_{d=1}^{D} X_d^{n,i}. \tag{19}$$

This mean is then calculated for each value in each sample. The term for the variance of our pixel values through all realizations is then:

$$\mathrm{var}\left(n,i\right) = \frac{1}{D} \sum_{d=1}^{D} \left(X_d^{n,i} - \left\langle X^{n,i} \right\rangle\right)^2. \tag{20}$$

Given this variance between realizations, a general function for the covariance for any pixel with any other pixel in a given realization can be written as follows:

$$\mathrm{cov}\left(n,i;n',i'\right) = \frac{1}{D} \sum_{d=1}^{D} \left[\left(X_d^{n,i} - \left\langle X^{n,i} \right\rangle\right)\left(X_d^{n',i'} - \left\langle X^{n',i'} \right\rangle\right)\right]. \tag{21}$$

For the purposes of this work, it is important to know how correlated nearest neighbors are to one another. This is because they have the highest 'risk' of being sampled prematurely. To systematically measure these correlation the following covariance values are important:

$$\mathrm{cov}\left(n,i;n+1,i\right) = \frac{1}{D} \sum_{d=1}^{D} \left[\left(X_d^{n,i} - \left\langle X^{n,i} \right\rangle\right)\left(X_d^{n+1,i} - \left\langle X^{n+1,i} \right\rangle\right)\right]. \tag{22}$$

Given the above, the formulation to calculate the correlation is as follows:

$$\mathrm{corr}\left(X\right) = \frac{1}{I}\frac{1}{\left(N-1\right)} \sum_{i=1}^{I}\sum_{n=1}^{N-1} \frac{\mathrm{cov}\left(i,n;i,n+1\right)}{\sqrt{\left(\mathrm{var}\left(i,n\right) * \mathrm{var}\left(i,n+1\right)\right)}}.$$

Two tests were done to make sure that the code was working as anticipated. The first is by the use of anti-correlated data. That is, what is the likelihood that an observation will be the opposite of what was observed before. To test this, data was contrived which alternated between matrices of fully zeros and fully ones. For each realization of this the order switched. Results outputted 0.999999995 which was to be expected from a working algorithm.

Next, Ising data generated by Monte Carlo algorithm was taken during several runs parameterized by the decorrelation time or the amount of Monte Carlo steps between each recorded data point. Figure 7 shows the correlation as a function of the decorrelation time.

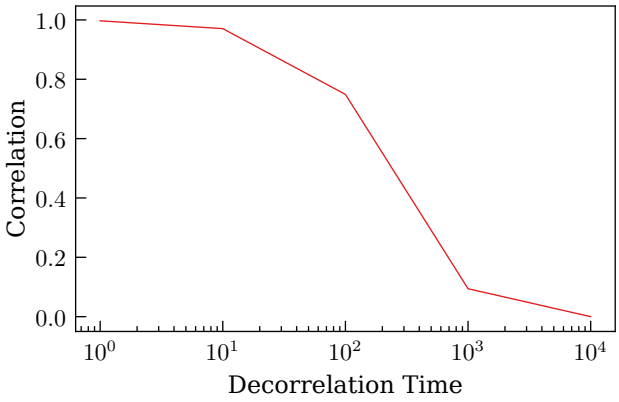

Figure 7. Plotting correlation values as a function of decorrelation times for 64x64 lattice Ising model parameterized with T=3.5. The decorrelation plateaus at the end of the graph where the final point was just over $2 \cdot L^2$.

VI. THE ISING MODEL: NONEQUILIBRIUM

A. Glauber Dynamics

The Ising model is by nature probabilistic and can exhibit dynamic like properties. What needs to be done is to explicitly define a time step. An intuitive and common definition for a single time step in simulated dynamics for the square Ising model is L^2 Monte Carlo attempts at flipping a site where L is the side length and L^2 is the number of sites. Saving samples and marking them with an evolving time signature allows one to observe dynamics both in and out of equilibrium, examples of these dynamics can will be presented later in the work; for example in figure 21 [60].

The algorithm is carried out in the following way:

- Start from some initial configuration with L^2 spins.
- For every time step, repeat the following L^2 times:
  - Pick a random spin and compute the energy difference ΔE for flipping it.
  - Flip the spin with probability $\frac{e^{-\beta \Delta E}}{1+e^{-\beta \Delta E}}$ (i.e. enforce detailed balance).

Where L is the side length of a square lattice with L^2 spines. ΔE and β are the same as those outlined above.

By defining a time step as L^2 Monte Carlo steps, the dynamics can be mapped as frame by frame snapshots, taken one time step apart from the next.

B. Nonequilibrium Glauber Dynamics

With an unambiguous definition for time in which to portray dynamics, equilibrium and nonequilibrium dynamics can be studied. This is true even though the Hamiltonian describing the Ising model, and the density of samples, are defined for in equilibrium conditions. The reason for this is that at any given time step, the effective conditions acting on the sample can be taken as equilibrium conditions [60].

The Hamiltonian of the Ising model described in equation (13) has a term h which adds a spin bias. This bias can be viewed as an effective magnetic field. This magnetic field, h, can be parameterized by time variable defined by Glauber dynamics t as follows:

$$h\left(t\right) = H\sin\left(\omega t\right). \tag{23}$$

Where H is the magnitude of the magnetic field and ω defines the frequency.

Equation 23 along with the temperature gives a rich parameter space to explore dynamics. Additionally, given the cyclic nature of the the oscillating magnetic field, each oscillation can be view as a self contained experiment. This supplies extremely inexpensive experimental data for investigation. The fruits of this exploration will be shown in the results section.

Methods

VII. CONTINUOUS SPACE DENSITY ESTIMATION

A. Increasing Complexity

Unfortunately, this simple Monte Carlo sampling method quickly collapses when the dimensionality of the problem increases.

The issues become apparent even when slightly increasing the complexity of the distribution of interest. Moving from 1-d to 2-d will immediately display the approaches shortcomings.

Suppose samples are drawn whose values fall within a range of 0.0 - 10.0. The space is broken into ten bins of equal length. With 100 samples taken we would have an average of 10 points per bin. Now considering the two dimensional case, to reach the same level of resolution 100 bins would now need to be filled. If the same amount of data was used as in the 1-d case, there would only be, on average, one point per bin. Either resolution would need to fall by a factor of almost three in each dimension or 10 times the amount of samples would need to be generated.

Finding ways to circumvent this particular "curse of dimensionality" is crucial for estimating the entropy of more complex systems.

B. Entropy and Independence

It turns out that distributions comprised of multiple independent random variables don't pose a problem to the calculation of entropy. This is because the entropy of such distributions can be calculated by taking the sum of the two values of entropy of its component random variables. For a distribution P with independent variables x and y the entropy can be written as:

$$S(x,y) = -\int\int P(x,y)\ln(x,y)\,dxdy$$

$$= -\iint P(x,y)\ln(x)\,dxdy - \iint P(x,y)\ln(y)\,dxdy$$

$$= -\iint P(x,y)\,dy\ln(x)\,dx - \iint P(x,y)\,dx\ln(y)\,dy$$

$$= S(x) + S(y). \tag{24}$$

This holds true for any number of random variables given they are independent. This property of independence and entropy will act as an important foothold for navigating the problems of entropy calculation in higher dimensions.

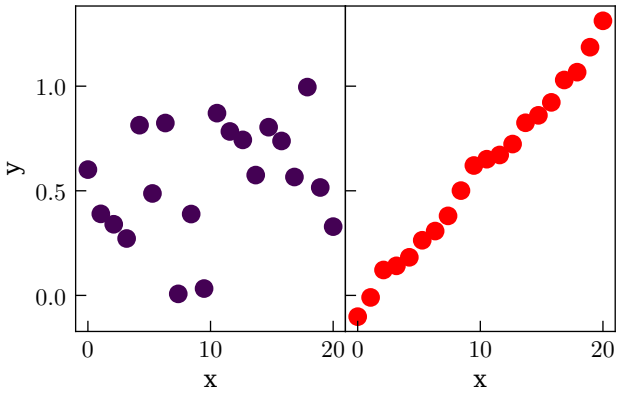

Figure 8. Examples of distributions with low (left) and high (right) correlation plotted on arbitrary axes x and y.

C. Linear Correlation

Nature is highly interconnected and systems are often comprised of many dependent random variables. That said, there exists patterns of dependence which can be utilized to allow for a simplification in the difficult task of calculating entropy. One such pattern is the degree of linear correlation.

A system can be described as linearly correlated when an increase of one variable correlates to a linearly consistent change in a second variable. When an increase in one variable is mirrored by an increase in another then the linear correlation is said to be positive. If, instead, an increase of one variable is met with a decrease in the other than the variables would have a negative linear correlation.

In figure Figure 8 on page 25 there are two simple examples of data with low, left panel and high, right panel, linear correlations. The entropy of the uncorrelated example can be calculated by summing the entropy of x and y independently. This is not the case for the lineally correlated model.

D. Principle Component Analysis

For any linearly correlated distribution there exist a vector which best represents the correlation of its data. We can define such a vector as one which, when data is orthogonally projected onto it, maximizes the variance. It can also be said that it is the vector which

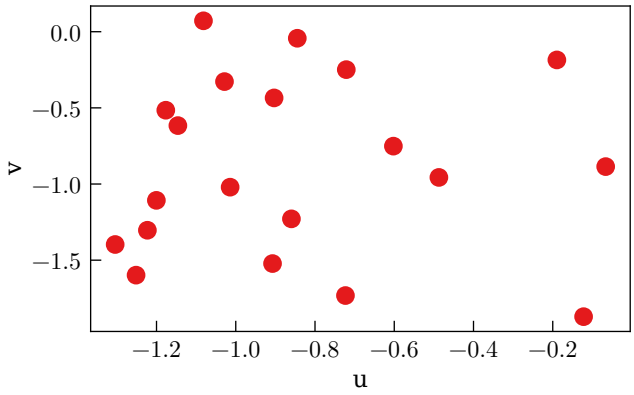

Figure 9. Result of using principle component analysis on the 2d linearly correlated data set in Figure 8 on page 25b. The graph now strongly resembles the uncorrelated data found in 8a

minimizes the mean-squared difference between the data and their projected values on to such a vector [61]. This vector is called the first principle component. There is a principle component for every dimension of the sample data, all of which are orthogonal to each other and all of which decrease in magnitude from one to the next.

The principle components can be found by calculating the eigenvectors of the distribution's covariance matrix. Computing a linear transformation in which the data set is multiplied by these eigenvectors will set the principle component vectors as the new basis vectors. This new, transformed, data set will be the most accurate approximation to an uncorrelated distribution.

Figure Figure 9 on page 26 is what happens after preforming a principle component analysis on the linearly correlated data in Figure 8 on page 25. This new, linearly transformed, data set gives an entropy similar to the original when adding the x and y components and costed much less computational time to arrive at that answer.

E. Kernel Density Estimation

Often times systems have correlations that are not linear. In such scenarios PCA will yield vectors which will be unhelpful in understanding the nature of the correlations.

Kernel Density Estimation (KDE) is a more flexible tool which provides a means of extrapolating density from limited sample observation. The kernel, for which the name arrives, is a continuous, non-negative, function where the value of the sample X is a constant

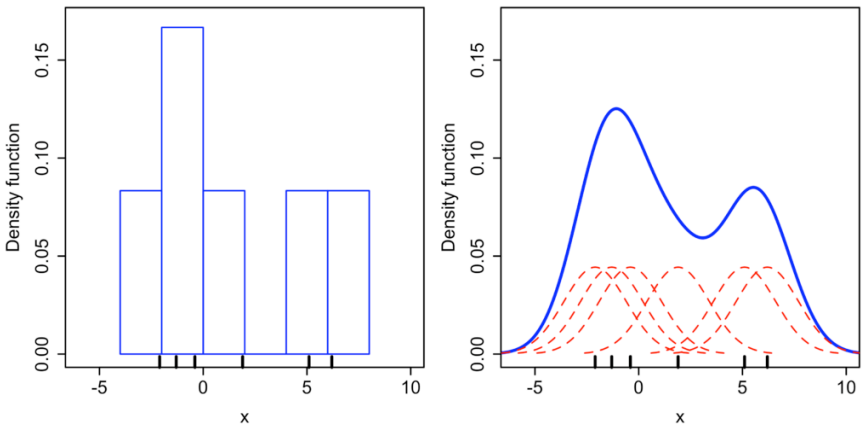

Figure 10. Given six observations of a one dimensional distribution two algorithms are being used to represent the presented data; histogram (left) and kernel density estimation using a normal function as the kernel function (right). The kernels act as a way to buffer the bias of limited data and generate much smoother estimations for the distributions [62].

in the function. For example,

$$P_{\mathrm{KDE}}(X) = \frac{1}{n}\sum_{i=1}^{n} K_h(X - X_i). \tag{25}$$

The functions have a smoothing parameter, h, which is often referred to as the bandwidth. The bandwidth parameter is tuned so that the minimum value can be found to accurately represent the distribution. When tuning the bandwidth there is a trade off between bias and variance. These functions can assume a wide range of values and the dimensionality of them can vary with the needs of the estimation task. A common kernel is the Gaussian kernel,

$$K = e^{\frac{-(x_i - x)^2}{2h}}, \tag{26}$$

where the sample, x_i, represents the well-defined mean of the function and the bandwidth, h, is the standard deviation.

It is common to parameterize the bandwidth by the standard deviation of the data set. This gives sharper functions when the data has lower deviations and broader functions when deviations are larger.

In Figure 11 on page 28 a Gaussian distribution is shown as well as a collection of samples drawn from it. Using that sample data, a KDE was used to generate two distributions. In one the bandwidth parameter was set too low, this created a model where the standard

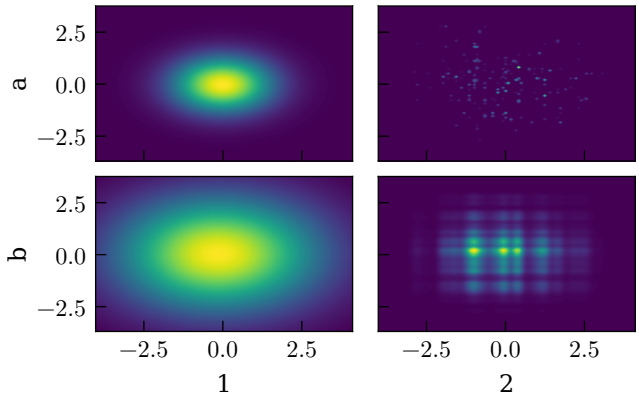

Figure 11. Representation of a Gaussian distribution, subplot(a,1) and one hundred data points placed into a histogram, subplot(a,2). The bottom two subplots represent kernel density estimation using first a bandwidth which is too low in subplot(b,1) and the other which is not low enough in subplot(b,2).

deviation is much larger than that of the actual distribution. In the second example, the bandwidth was not set low enough. The bias ended up being much too strong in this case and the distribution barely even resemble a Gaussian. In theory, the KDE should be able to build an extremely good model given that the kernel is the same function as that which was trying to be represented.

VIII. ISING MODEL DENSITY ESTIMATION

A. Lossless Compression Algorithms

In the case of the Ising model, the number of states grows exponentially with an increase in system size at a rate of 2^N where N is the number of sites [14]. This necessitates the need for methods of estimation which only need a small fraction of samples to represent a much bigger space.

When approaching the task of measuring entropy it can be helpful to view the concept of entropy from all formulations made in the past. Information theory, born out of the work by Shannon [63] and Kolmogorov [40], drew mathematically identical lines between the statistical-mechanics definition of entropy and the informational entropy at the limit of large data. Shannon defined the entropy as the average information gained per observation

of a system. This is in a scheme where the means for communicating the information of our system is ideal. In this way the entropy is a measure for the limit of lossless compression.

Lossless compression algorithms work to realize the theoretical concepts set out by Kolmogorov's formulation of complexity.

B. Lempel and Ziv algorithms

Lempel and Ziv (LZ) laid out schemes which ended up being the most adopted approach for implementation of lossless-compression algorithms [64], [65] [66].

The algorithms work by receiving data sequentially and processes that input and replacing repeating segments with pointers to past instances of that segment. The data must belong to a finite alphabet. Given enough data it has been shown that the ratio of LZ compressed data to the raw input sequences converge to Shannon's definition of entropy [67][68].

C. Autoregressive models

For any set of independent and equally distributed random variables, the likelihood of an event has no dependence of what happened before it. In this way the probability of two events, A and B, occurring is the same as the product of those two events happening independently. This extends to the following equality:

$$P\left(A|B\right) = P\left(A\right), P\left(B|A\right) = P\left(B\right), P\left(AB\right) = P\left(A\right) P\left(B\right). \tag{27}$$

For configurations assumed by the Ising model, samples can be viewed with each site being a random variable. If there were no correlations in the data, between sites, then the probability density of a given configuration would be:

$$P\left(X\right) = \prod_{i=1}^{n} P\left(X_i\right), \tag{28}$$

where n equals the number of sites.

However, it is clear that there are correlations between Ising model data; ;ow temperatures samples are almost completely correlated as the model assumes a fully ferromagnetic state.

It is intuitive to look at time as an axis on which to find correlations. This is especially true for natural processes which are known to evolve over time. Statistical formulations

on how to implement this approach go as far back as the 1920's with the introduction of the first auto regressive models [69]. These models are feed-forward models in which the density of later observations are constructed as conditionals from the earlier observations. For long strings of events, events that appear later will be dependent on a lot of data which came before it. This rich source of information is the basis for how the PixelCNN machine learning model parameterizes its networks and works as an effective density estimator.

D. Applications towards Entropy Estimation

There are multiple approaches to minimize the mutual information between increasingly large subsystems using artificial neural networks [70] [71] [72].

The method proposed by this work attempts to estimate the entropy by first performing a different, seemingly harder task: density estimation. With the probability density and samples from a simulation available, evaluation of the entropy becomes trivial. This approach has been proposed with some success [73]. In particular, we applied PixelCNN++, an autoregressive density estimation algorithm originally designed for image processing [74] [75], to the Ising model on a 2D square lattice. This is a canonical model for magnetism where analytical results are available [49] in a certain parameter regime, and therefore a natural benchmark also used by previous methods. We showed that the approach performs well at a variety of system sizes, then demonstrated its usefulness by exploring the role of entropy in non-equilibrium Glauber dynamics describing the behavior of an Ising model within a rapidly oscillating magnetic field.

E. PixelCNN

PixelCNN [74] [75] works by leveraging the chain rule in order to decompose the likelihood of a sample $\mathbf{x}$ into a product of 1-d distributions where each pixel is its own neural network. Each sample is broken down into its composite pixels and the model is trained by defining the density of the i-th site as

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}) . \tag{29}$$

Using this chain of conditional probabilities, PixelCNN can define a likelihood function to act on each pixel and train it against our training data. For every site we can construct a distribution which is dependent on the sites which came before it. These joint probabilities are then maximized to the log-likelihood that a pixel will resemble our training data and formulated as regressive functions parameterized over the pixel values which came before it.

Decorrelated Monte Carlo samples generated from the Metropolis-Hastings algorithm act as the ground truth in the training process. This, in effect, breaks down the complex problem of finding the full distribution into many different classification problems, site by site, where each step is built from the conditional probabilities of steps which came before it. The result is a fully tractable probability distribution which can estimate the entire sample space.

The auto-regressive model is one that creates functions based solely on the actual distribution, using the chain rule of joint probability distributions. A three dimensional vector will provide a simple example with which to articulate this exactly:

$$\boldsymbol{x} = x_1, x_2, x_3; D = 3,$$

$$P_m(\boldsymbol{x}) = P(x_1)\, P(x_2|x_1)\, P(x_3|x_1, x_2)\,. \tag{30}$$

We can write all of these probabilities as parameterized functions which can essentially be linear equations parameterized by weights and biases. These functions can be the input to some other type of function which will restrict its value to something between 0,1.

$$Sigmoid : f(x) = \frac{e^x}{e^x + 1}, \tag{31}$$

$$P(x_1) = P_1 = f\left(b_1^1\right) = F_1(\theta_1)\,, \tag{32}$$

$$P(x_2) = P_2(x_1) = f\left(W_1^2 x_1 + b_1^2\right) = F_2\left(x_1; \bar{\theta}_2\right), \tag{33}$$

$$P(x_3) = P_3(x_1, x_2) = f\left(\sum_{i=1}^{2} W_i^2 x_i + b_2^2\right) = F_3\left(x_1, x_2; \bar{\theta}_3\right), \tag{34}$$

30 has the composition of a product so it will be easier to calculate by taking the logarithm:

$$\log\left(P_m(\boldsymbol{x})\right) = \sum_{d=1}^{D} \log\left(F_d\right). \tag{35}$$

This provides a parameterized equation for the actual distribution. The next step is to define a loss function to train a model on. A convenient function is the Kullbeck-Leibler Divergence [7] [22] because it is a measure of a distance between two distributions and because that distance is zero when the two distributions are the same, thus making it obvious that minimizing the function will give us usable values for our weights. It is written in the form:

$$D_{\mathrm{KL}}\left(P \| Q\right) = \sum_{x \epsilon X} P\left(x\right) \log\left(\frac{P(x)}{Q(x)}\right). \tag{36}$$

In order to utilize the KL Divergence it is necessary to have a reference to the ground truth with which to compare the distribution:

$$P_{\mathrm{real}}\left(x_1, x_2, x_3\right) \rightarrow \left(x_1^s, x_2^s, x_3^s\right); \left[s = 1, .., N\right]. \tag{37}$$

Sample data taken from a distribution is often what is used, acting as an incomplete but useful reference. A delta function is created which is often refered to as the prior distribution and is defined like:

$$P_{\mathrm{prior}}\left(x_1, x_2, x_3\right) = \frac{1}{N} \sum_{s} \delta\left(\boldsymbol{x} - \boldsymbol{x}_s\right). \tag{38}$$

The value of the function is $\frac{1}{N}$ wherever we have an observed sample.

Using this as a reference point a comparison to the models distribution to the prior,

$$D_{\mathrm{KL}}\left(P_{\mathrm{prior}} || P_{\mathrm{m}}\right) = \int P_{\mathrm{prior}}\left(\overrightarrow{x}\right) \log\left(\frac{P_{\mathrm{prior}}(\boldsymbol{x})}{P_{\mathrm{m}}\left(\boldsymbol{x}\right)}\right) d\boldsymbol{x}. \tag{39}$$

The property of logarithms can be used to split the division term inside the logarithm into two, one of which is independent on P_m and is therefore a constant value. Because the absolute value of the KL Divergence is uninteresting and instead the function just needs to be minimized, that term can be ignored. The $\frac{1}{N}$ term from 38 will also be ignored leaving:

$$D_{\mathrm{KL}}\left(P_{\mathrm{prior}} || P_{\mathrm{m}}\right) = -\frac{1}{N_{\mathrm{s}}} \sum_{s} \int \delta\left(\overrightarrow{x} - \overrightarrow{x}_{\mathrm{s}}\right) P\left(\overrightarrow{x}\right) = -\sum_{s} \log\left(P_{\mathrm{m}}\left(\overrightarrow{x}_s\right)\right). \tag{40}$$

This design allows makes it convenient to leverage the rule of logarithms to free ourselves from the curse of dimensionality. The terms are now linearly dependent and therefore steps along the loss function and move down it step by step, batch by batch.

F. Generative models

A generative model is a model of $P(X)$ that can also generate configurations with respect to $P(X)$ Germain *et al.* [76].

The trained PixelCNN is also a generative model. The ability to generate samples with respect to $P(X)$ acts as an effective means to quickly asses that the model has learned the general "idea" by comparing generated samples to the training samples.

G. Re-purposing Tensorflows PixelCNN++ algorithm.

TensorFlow library has a working model for PixelCNN called PixelCNN++ [77]. The plus signs represent computational optimizations leaving the essential method unchanged.

The library was designed to take 2-d images with gray scale pixel values as inputs. With relatively little tinkering the algorithm was repurposed to accept 2-d Ising model samples.

H. Training and validation loss

When training neural networks there are many hyper-parameters, also known as network parameters, which are set before the model is trained. These parameters are not changed through the course of the training unlike the weight parameters which are changed using algorithms like gradient descent. While there are algorithms being used to automatically optimize the selection process of hyper-parameters, this work does not utilize those findings [78]. There are still hyper-parameters that must be tuned to avoid making obvious mistakes in the training of a model.

Figure 12 on page 34 presents a common diagnostic task of plotting the training loss as a function of the epoch and the validation loss as a function of the epoch. The training loss is the value of the function that the model is attempting to reduce through gradient descent, discussed above when explaining the KL divergence. The validation loss is a metric

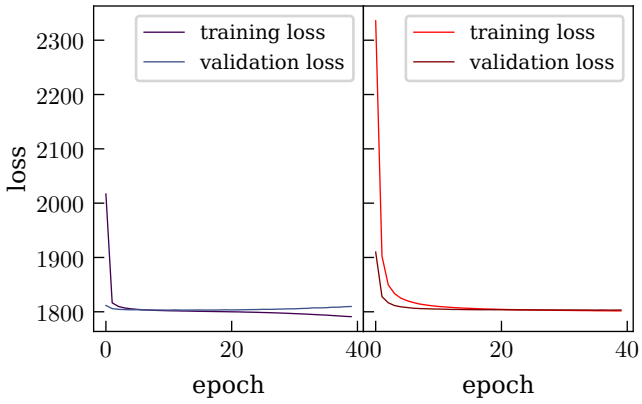

Figure 12. Showing the training loss (often just referred to as the loss) and validation loss per epoch of two separate models which differ only in the training rate. (a) Left: model with a training rate of 0.0005 which ended up being too large for the amount of epochs chosen. (b) Right: model with a lower training rate of 0.0001 which converged nicely to the test loss, showing proof the model did not over fit the data.

to determine how well the network performs on new data.

Before training begins, the available data set is broken up into two groups, the training set and the validation set. The training set is shuffled and reused every epoch, acting as the ground truth through the training process. At the end of each epoch the network is shown the data from the validation set and the average loss per sample is recorded. Gradient descent is not done at this point, it is important to not train the model on the validation set in order to preserve it's integrity as a diagnostic tool.

Figure 12 on page 34 is plotting the loss as a function of the epoch for the same system with only a difference in the learning rate. The plot on the left shows a case where the learning rate, the magnitude at which gradient descent makes its incremental steps, is set five times higher than the network presented on the right. At around epoch number twenty, the test loss continues to decrease while the validation loss starts to increase. This is a clear sign that the model is over fitting the data. Contrasting that to the plot on the right, the lower learning rate allowed for congruent agreement between training and validation loss, showing that the model has yet to over fit. For the purposes of this work, where ample training time and computational resources are available, it is more important to not over fit the data than to hastily attempt to find convergence.

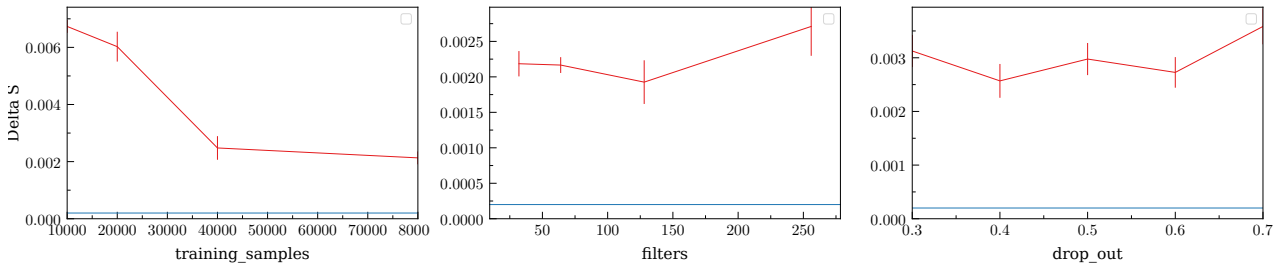

Figure 13. Explores influence of specific hyper parameters on performance. Models were trained and tested on their ability to accurately estimate the entropy of the 2-d Ising model in equilibrium at the critical temperature. Working from the left: the number of training samples used per epoch showed to have an increased performance as the amount of data was increased. Increasing the number of filters (middle) had diminishing returns as the model likely began to over fit the data. Finally the drop out was increased (right) and failed to yield reliable results for this test. It is worth noting that drop out did increase the accuracy of the model when more filters were used.

I. Optimizing Hyperparameters

While this work is not making claims to have fully optimized the tuning of hyper parameters, a screening process was to observe their effects on the model. The simplified search was done by changing a single parameter though a given range of values and testing the effects on the final resulting entropy which that had.

In addition Keras is a framework built on top of Tensorflow 2 which has a large collection of callback functions to assist in the training of models. These proved extremely useful in that they were able to save weights only when an improvement in the training took place, in our case when the validation loss went down. For large models trained over many epochs, this helped reduce the amount of storage tremendously. Built in functions were also able to end training after a specific number of unimproved training epochs passed. This made training to saturation much simpler,

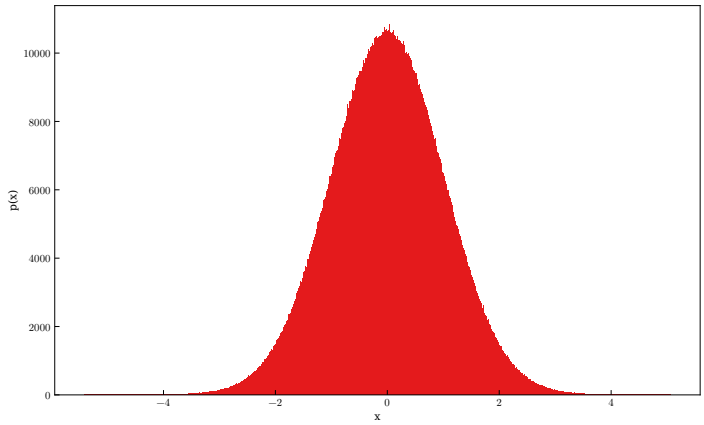

Figure 14. $2 \cdot 10^7$ samples generated from a Gaussian distribution by metropolis algorithm and binned into $7 \cdot 10^3$ bins. Because of the low dimensionality of the distribution under investigation, the histogram method works very well in building a reliable model

Results

IX. CONTINUOUS SPACE DENSITY ESTIMATION

A. 1-Dimensional Density Estimation using histogram

A 1-dimensional test case of a Gaussian distribution was developed. Using the Metropolis-Hastings algorithm, a data set reflecting a standard Gaussian distribution was generated. The probability density function for this distribution is:

After the data set was created, an appropriate range was chosen on the x-axis and the space within that range was divided into bins. Counting the number of points that fell in a given bin interval gives the bins its value, represented graphically by its height. $2 \cdot 10^7$ points were drawn from the above distribution are presented in figure 14 with $7 \cdot 10^3$ bins. It is plain to see both the familiar shape of the normal distribution while also seeing the noise caused by limited sampling.

Given the nature of the data being composed of a known number of samples, the distribution can be easily normalized. This is done by taking the output probability estimates, $p(x_i)$, for each bin and then dividing that value by the total number of samples. The entropy can then easily be calculated:

$$-\sum_{i=1}^{n} p(x_i)\ln(p(x_i))\Delta x, \tag{41}$$

where bin value is $[x_1, ..., x_n]$ and Δx is the size of each bin. In the limit of infinite data and infinite bins, the above equation approaches the true value for the entropy of this distribution. In such a case the summation term is written as an integral:

$$-\int Ae^{-\alpha x^2}\ln(Ae^{-\alpha x^2})dx, \tag{42}$$

$$-A[\ln(A)\int e^{-\alpha x^2}dx - \alpha\int e^{-\alpha x^2}x^2dx]. \tag{43}$$

Using the symbolic computer python library, Sympy, the analytic entropy was calculated. The numeric approach shown above, which took under a minute of computation time on a local machine, generated samples reflecting a distribution with a ΔS of 0.00078.

This numerical approach is a fast and reliable method for approximating the entropy for certain, very simple, distributions.

B. Combining PCA with histogram density estimation

To prove the viability of the PCA method, samples were drawn from a rotated Gaussian distribution. The distribution is defined as:

$$a = \frac{\cos(\theta)^2}{2\sigma_x^2} + \frac{\sin(\theta)^2}{2\sigma_y^2},$$
$$b = \frac{-\sin(2\theta)^2}{4\sigma_x^2} + \frac{\cos(\theta)^2}{4\sigma_y^2},$$
$$c = \frac{\sin(\theta)^2}{2\sigma_x^2} + \frac{\cos(\theta)^2}{2\sigma_y^2},$$

$$P(\mu_x, \mu_y, \sigma_x, \sigma_y, A) = Ae^{-a(x-\mu_x)^2+2b(x-\mu_x)(y-\mu_y)+c(y-\mu_y)^2}. \tag{44}$$

Where μ and σ are the means and standard deviations per axis and θ is the angle at which the rotated axis has with respect to the x-axis. A is the magnitude of the distribution. For the simulations done in this work the mean values were set to zero and the standard deviations and magnitude were set to 1.0; θ was set to π.

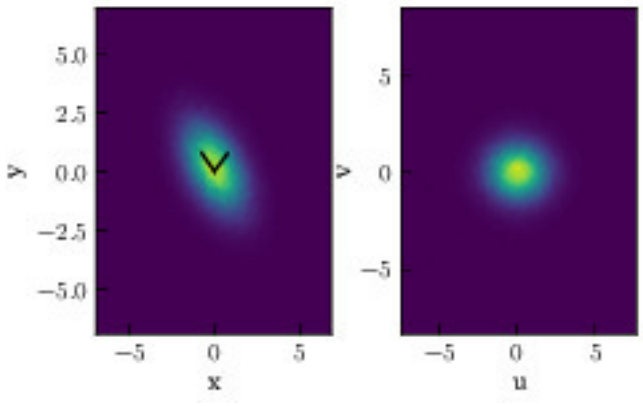

Figure 15. Rotated Gaussian distribution before (a) and after (b) PCA guided linear transformation using Monte Carlo 1,000,000 generated samples and histograms 100x100 bins. The linear correlation is almost fully erased under the new basis vectors.

This type of distribution is attractive because the analytical value of its entropy can be calculated easily. The reason for this is because the distribution is comprised of a product of two Gaussian distributions, making it completely linearly correlated.

In 15 (a) a two dimensional histogram is shown with data generated using the Metropolis-Hastings algorithm. PCA was done and the first two principle components are highlighted in black. The distribution was then transformed onto the principle components as it's new main axes, u and v, and re-scaled to exhibit the symmetry seen in figure 15 (b).

From this new transformed data set the entropy of the original distribution can be calculated as though it were a product distribution of two Gaussian distributions. For this specific example, where the function with linear dependence is truly a Gaussian, we know with enough samples the system will converge perfectly with the analytically calculated entropy.

C. Tricking Principle Component Analysis

Understanding the limitations of a technique is important. Some distributions exhibit Gaussian properties radially but not in Cartesian coordinates. Figure 16 (a) shows a distribution which resembles a halo which can be defined mathematically as:

$$P\left(\mu_x, \mu_y, \sigma_x, \sigma_y, A\right) = Ae^{-a(x-\mu_x)^2+c(y-\mu_y)^2}, \tag{45}$$

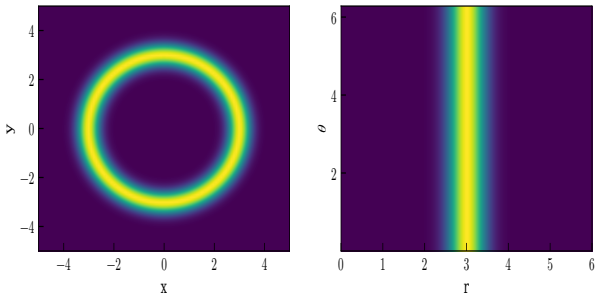

Figure 16. Example of a distribution with radial symmetry (left) and its transform to a radial coordinate system (right). PCA would fail to find meaningful principle components in the radially symmetric distribution.

in Cartesian coordinates and

$$P(r, \theta, \sigma) = e^{\frac{-(r-R)}{2\sigma}} \tag{46}$$

in radial coordinates.

Performing PCA on such a distribution would be meaningless as, at the limit of infinite samples, any direction is equally likely for the first principle component. The same distribution viewed in radial coordinates (figure 16 (b)) makes its Gaussian properties more obvious. The provided graph could be reduced a dimension because theta doesn't effect the distribution. A histogram with intervals of radius length would be more easily recognizable as Gaussian.

Another example of a set of distributions that would fail to be accurately described by PCA are distributions that are equally balanced on multiple different axes where only one of which reflects accurately the product distribution. In figure 17 (a) we have a representation of the data from samples drawn form a Metropolis algorithm equipped with long jumps to escape the large distance between wells. It can be seen where the four axes of symmetry lie, all of which are equally likely for the PCA algorithm to chose. Figure 17 (b) and (c) represent two equally likely scenarios for the transformation around the first two principle components. Increasing the dimensionality of the problem would allow for more areas of symmetry, further accentuating this weakness of the algorithm to accurately identify the principle components in which would allow for the reduction of dimensionality and the estimation of entropy.

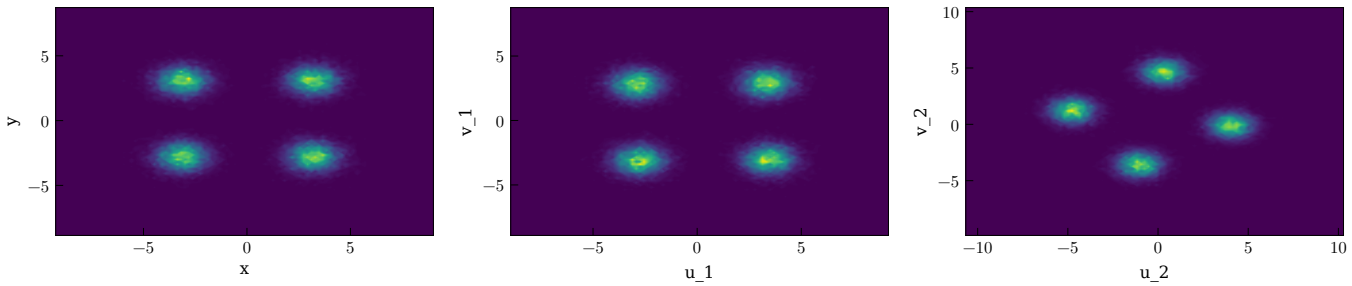

Figure 17. Examples of distributions that are highly sensitive to imbalance, rendering PCA ineffective. (a) represents 60,000 samples drawn from a distribution composed of four smaller Gaussian distributions using the Metropolis-Hastings algorithm. (b) and (c) represent the two equally likely possibilities of transformation using principle components as new basis vectors.

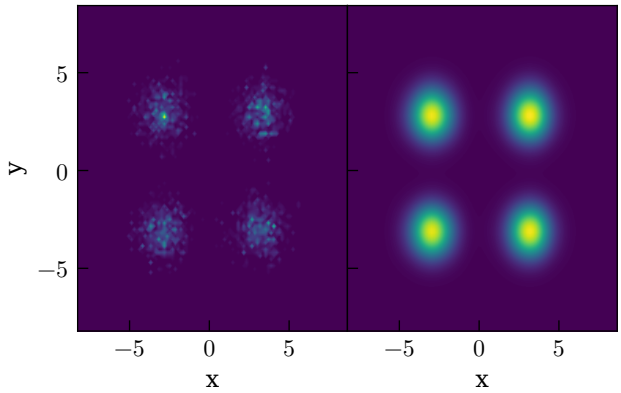

Figure 18. Example of how the distribution found in Figure 17 on page 40a estimated using a KDE.

D. Overcoming limited sample size

Other methods of density estimation would subvert the possibility of getting false positives in the data. In figure 18 the same distribution which tricked PCA was effectively measured using KDE. Some trial and error was needed to move through the bandwidth parameter space to find a reasonable value.

Choosing the right kernel and bandwidth is not always simple. Knowing the distribution before hand made the process trivial but in cases where a benchmark is not known, other, more complicated methods must be developed to properly investigate which parameters allow the model to represent the distribution most accurately [79].

In this case, because of the Gaussian nature of this distribution, it was no shock that the

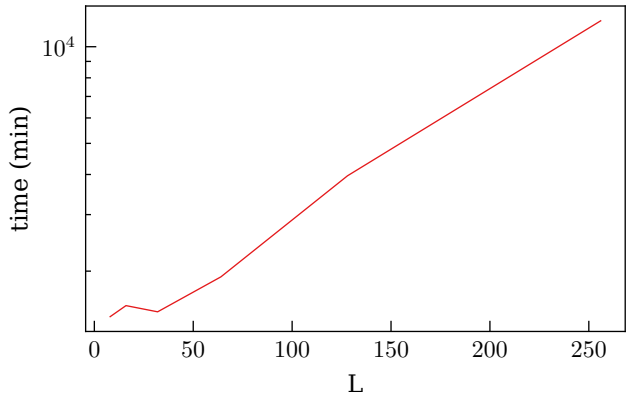

Figure 19. Training time as a function of the system size where L is a side length of a square lattice.

resulting KDE produced accurate results.

X. WORKING ON HIGHER DIMENSIONAL OBJECTS

A. Analytic bench-marking of PixelCNN

The methods explored above did manage to yield satisfactory results but were strongly limited in their ability to handle multidimensional objects. This was shown in the ways that basic histograms require too much data and PCA could be tricked by certain radial symmetries. KDE, in theory could be generalized, but the methods parameters often cannot cope with the multidimensional correlations of the distribution that likely do not behave like the chosen kernel [80]. Instead a method that is both flexible and robust is needed which can account for the diverse and long range correlations that might exist.

The PixelCNN method was used on Monte-Carlo generated samples of the Ising model at various different system sizes. The PixelCNN network was trained on those samples as its ground truth. Networks were trained on 10,000 training samples and tested on 1,000 samples. Figure 19 shows the effect that system size has on the training time of the network. The lattice of $L = 32$ exhibits both strange behavior in the training time as well as the accuracy of the model, the reasons for this are yet unknown but an investigation into the parameter space is being made in order to either break the performance of reproduce it in other system sizes.

Figure 20 shows the performance of the trained network for samples in equilibrium. The network is a generative one meaning it is able to generate samples with respect to the model distribution. This capacity is illustrated in the stack of panels on the right side of figure 20. There samples generated by PixelCNN are compared to the Monte Carlo simulated samples for a range of temperatures. Before even needing to look at the analytic benchmark, it is reassuring to see that the samples generated by PixelCNN resemble those pulled from the actual distribution.

The left side of figure 20 has two panels displaying how the model performed against the analytic benchmarks. The top panel shows ΔS as a function of temperature for various different system sizes. ΔS in this case is the analytic result for the Ising model in equilibrium [81] subtracted by the entropy generated by the model.

The entropy is calculated for the model by feeding in Monte Carlo generated samples and having the model calculate the log-probability of each sample. The reiterate again, the overall model is in actuality a collection of many models, one for each pixel. The model for each pixel are built in an autoregressive fashion where the inputs to the model of any given pixel are the pixel values of those which came before it. The loss function used for all of these models is the log loss. This is particularly convenient for calculating the entropy where the log-probability is needed. The product of each pixel's log probability yields the log probability of a given sample happening. That value is divided by the system size to give the value of the entropy per site. This was done and averaged over 1,500 samples giving a degree of confidence as well as error bars for the calculations presented in figure 20.

It should be noted that there is a spike in the error as we measure near the critical regime of the 2d Ising model which is $T \approx 2.67$. The PixelCNN model out performs the MICE algorithm for both high temperature cases and for calculations at the critical regime where correlations are least predictable [82].

It can be observed that for larger system sizes the accuracy of the method increases. The degree of which is clearly illustrated in the bottom left panel of figure 20. The average of error, $\langle \Delta S \rangle$, for all the calculated temperatures is plotted as a function of the system size. Monte Carlo error scales like $\frac{1}{\sqrt{N}}$ where N is the number of samples [10]. Given that the amount of information contained in an Ising model sample can be viewed as a function of the number of sites, which is L^2 in this case, an increase in accuracy should scale 1-to-1 with system size. The fact that a less steep increase is observed probably has something

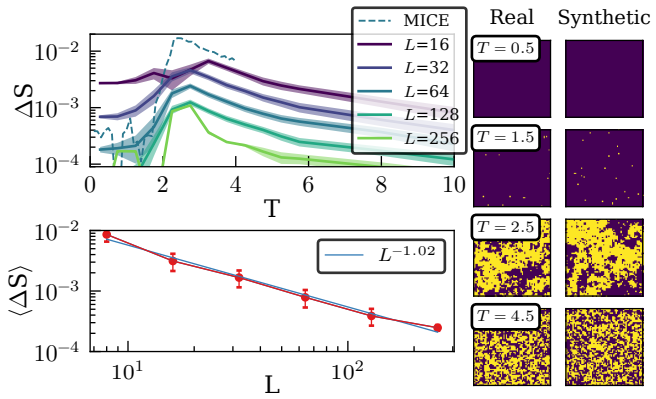

Figure 20. (a) Top left: the average error in PixelCNN entropy estimation technique as a function of system size. Portrays the accuracy of the method for several system sizes and compares the results to outside work labeled by the dashed black line. (b) Bottom left: shows the mean error over temperature in entropy estimation as a function of system sizes. (c) Right two columns: displays Monte Carlo generated samples next to samples drawn from the PDF estimated by the PixelCNN network for several different temperatures.

to do with the fact that for larger systems, more models are introduced which leaves more room for error and noise. Experiments were conducted where the model was trained on more data and the accuracy of the model increased by two orders of magnitude at the cost of comparable increases in computation time.

B. Exploring non-equilibrium phase space

Monte Carlo can accurately simulate these dynamics because for each time step the energy is well defined. This is true for any configuration despite them being at different times and having separate effective magnetic fields acting on them. For the work done here, the period for the oscillating field, ω, was held constant at 50 time steps per cycle as found in 23. Given this ability to sample dynamics and the flexibility to explore the potential parameter space, namely T and H, a few characteristic types of behaviors started to appear.

21 shows a collection of samples ordered in a time series which are examples of these characteristic behaviors. Initialized at a random configuration (the top left panel for each of the three images), each consecutive panel is a configuration sampled after every L^2 Monte Carlo steps or in other words for every time step. Figure 21 on page 44 (a) shows para-

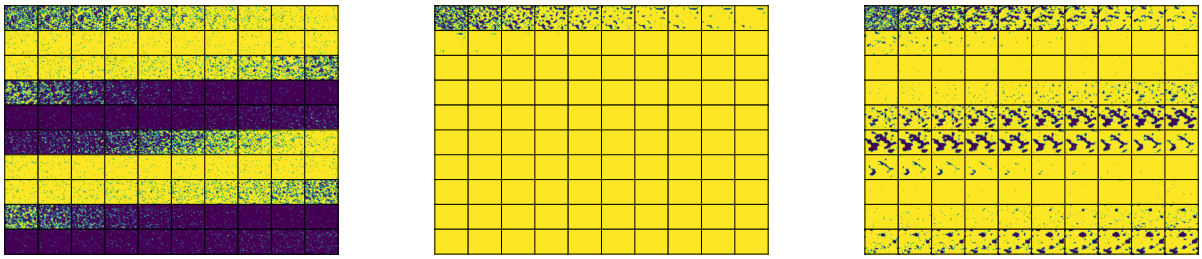

Figure 21. Characteristic behaviors which arise through investigation of the parameter space. Plots are composed of 100 frames of consecutive time steps during through a simulation of Glauber dynamics of an $L = 64$ square lattice, each under differentiating magnetic field strengths and temperatures. (a) The left most figure shows an example of dynamics within the paramagnetic regime, $H = 3$ and $T = 3$. (b) The middle plot is an example of dynamics within the ferromagnetic regime, $H = 0.5$ and $T = 0.5$. (c) The right most figure shows an example of dynamics within the chaotic regime, $H = 0.8$ and $T = 1.7$.

magnetic behavior where the mean magnetization of the model follows the direction of the magnetic field. This fluctuation happens dependently for every cycle of the field. Depending on the temperature and the field strength the response time of the magnetization of the model changes, either becoming more responsive, usually in higher temperatures, or slowly responsive and lagging behind the field, remaining saturated at a given magnetized state for most of the time. Figure 21 on page 44 (b) shows ferromagnetic behavior where, depending on the initial direction of the magnetic field, the system magnetizes towards the bias. After the system magnetizes it remains that way because the field strength is too low to disrupt the nearest neighbor effects on the energy of the sample. Figure 21 on page 44 (c) shows the most unusual behavior which arises on the border between the previous two regimes. Here the magnetic field is strong enough to influence the magnetization of the model but often not strong enough to totally flip it. In this way some long lasting correlation can begin to arise, surviving through potentially many cycles.

After these behaviors were observed by eye, order parameters became necessary to consistently and accurately define the behavior. The following order parameters were formulated to accomplish this goal:

$$O_{\text{ferro}} = \text{abs}\left(\text{avg}\left(M\right)\right), \tag{47}$$

$$O_{\text{para}} = \text{avg}\left(\text{abs}\left(M\right)\right) * \left(1 - O_{\text{ferro}}\right), \tag{48}$$

$$O_{\text{chaotic}} = \text{var}\left(M\right). \tag{49}$$

Where the order parameters for the ferromagnetic, paramagnetic and chaotic regime are O_{ferro} , O_{para} and O_{chaotic} respectively and M is the mean magnetization of samples.

The top three plots of Figure 22 on page 46 show the phase diagram constructed using the above mentioned order parameters. The level at which O_{ferro} was present added a blue color, O_{para} added a green color and O_{chaotic}a red color. It is worth noting that O_{chaotic} is formulated by taking the variance over many different realizations of the dynamics simulation. This order parameter also appears when the temperature is low and the magnetic field gets strong enough to cause fluctuations but not strong enough to flip the overall magnetization. So despite the red color on the bottoms of the phase diagrams, that is not the chaotic regime which is referred to in the rest of this work. It is also worth noting that the chaotic regime is larger in smaller systems, this is because the system is far more sensitive to noise.

The lower plots of Figure 22 on page 46 show, for each regime in the phase space, an example of the behavior of the mean magnetization with an overlay by the effective magnetic field. The mean magnetization, marked by the blue line, is an average value over 96 realizations of the dynamics simulation. For the case of the Chaotic regime, long time frames are presented to illustrate the varied behavior and unpredictability of the magnetization.

C. Exploring the chaotic regime

The phase diagrams were used as a reference to guide further investigations of the chaotic regime. Focusing on that region, for a given magnetic field strength, contour plots were made which depict the time averaged entropy and magnetization of the model.

Figure 23 on page 47 shows the results of this investigation. It should first be clarified how Figure 23 on page 47 was generated. Samples were taken for long run times over many cycles. Because of the cyclic nature of the magnetic field each cycle could be looked at as a

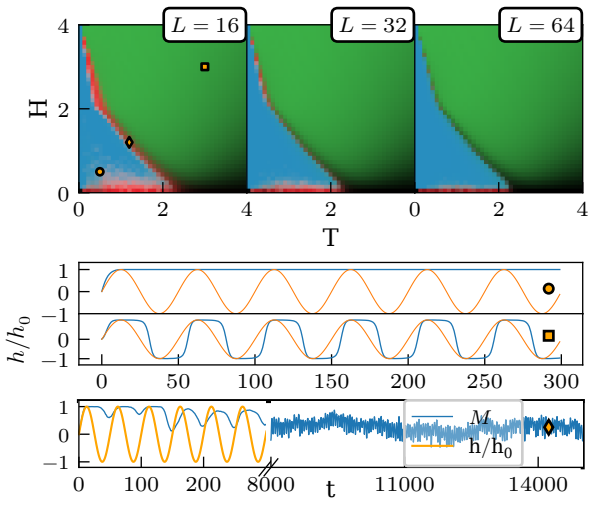

Figure 22. (a) Each pixel in the three phase diagrams (top) represents an Ising simulation run with parameters of magnetic field amplitude (H) and temperature (T). The colors show the prominence of order parameters. The three panels are system sizes $L = 8$, $L = 16$ and $L = 32$ (b) Examples of characteristic behavior of the Ising model: magnetized, linear response, saturated and chaotic; going from top to bottom.

stand alone experiment. The data broken up by cycle and was further separated into groups numbering the same as the amount of time steps per cycle, in this case fifty. So, for example, the first sample, the fifty first sample, the one hundred and first sample and so one would be put into one group. In this way all of the samples in each of the 50 groups all had the same external forces working on them. Then a PixelCNN network was trained for each of those data sets separately. This was done for several different temperatures ranging from 0.0 to 1.0 in increments of $\Delta T = 0.1$. The resolution was increased around the area of the chaotic regime, $T = 0.3$ to $T = 0.6$, from $\Delta T = 0.1$ to $\Delta T = 0.02$. Figure 23 on page 47 is therefore a contour of 22 temperatures and 50 time steps which totals to 1,100 uniquely trained models!

In the top right corner of Figure 23 on page 47 the mean magnetization is plotted for all of the data sets. This metric alone does not reflect the chaotic regime as well as the estimated entropy does. The zero mean magnetization acts as a clue to the increased state of the entropy but to quantitatively learn about the entropy is a different matter entirely.

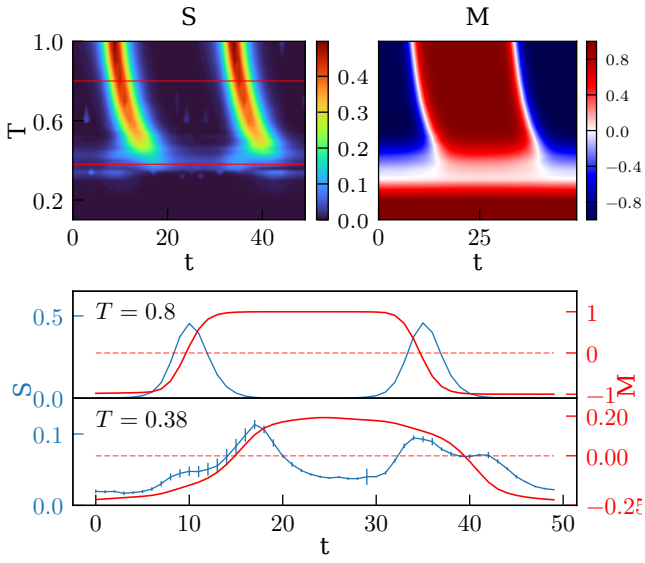

Figure 23. Describes the entropy and magnetization of the Ising model under an oscillating magnet field of strength $H = 2.5$. The top left panel shows time averaged entropy calculations for temperatures around the chaotic regime. The top right panel shows time averaged magnetization calculations in the same regime. The panel below shows a high temperature region outside of the chaotic regime displaying the time averaged magnetization (red line) and entropy (blue line) over a cycle. Just below that shows measurements for a temperature within the chaotic regime, showing higher relative entropy when the effective field is strongest as the system fails to fully magnetize during that time.

Such measurements of this regime is potentially unexplored physics of an old and well worn model. The one dimensional slices at the bottom of the figure show clearly the jump in entropy during a phase shift and the low entropy of the magnetized state outside of the chaotic regime and the relatively high entropy states when the slice is take at a temperature in the chaotic regime.

It is of interest to observe what happens to the regime as the system size increases. In Figure 24 on page 48 the results for models trained on four different system sizes, $L = 8$, $L = 16$, $L = 32$ and $L = 64$, are shown. It can be seen that the chaotic regime shrinks as the system size increases, getting concentrated to a smaller range of temperatures. Due to the number of models needed to be trained to produce these figures, and the increase in training time as the input data increases, the full extent as to whether this is just a finite

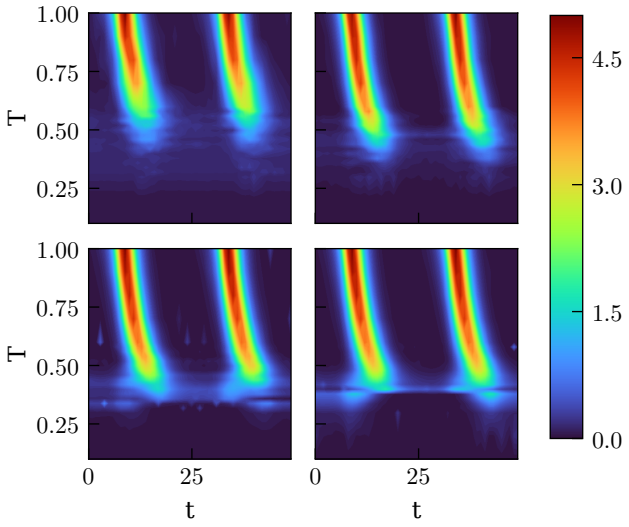

Figure 24. Shows contour plots of the PixelCNN calculated time averaged entropy for Ising models under the influence of an oscillating magnetic field, for a range of temperatures. System sizes are, from the top down and left to right, $L = 8$, $L = 16$, $L = 32$ and $L = 64$.

size effect has not been conclusively decided one way or the other.

Conclusion

Experimental data and simulations are our means of investigation to the behaviors and governing principles behind real world phenomena. For many systems the configuration space is extremely large and not tractable. For certain physical properties the full distribution is needed, one of which being the entropy. Utilizing symmetries and other assumptions of the system, it is sometimes possible to approximate the density of the system given a relatively small number of observations. However, it is often the case, that such assumptions are incorrect of unknown. In such cases, models which are both robust and flexible are of great use to theorist and sentimentalists alike.

In this work we presented a method which makes no assumptions onto the nature of the system and is capable of building models that are not restricted to equilibrium. The method proved viable by its accurate performance successful estimation of the entropy of large, high dimensional systems in the case of the 2d Ising model in equilibrium compared to its analytic benchmark. An experimental simulation was then carried out where the Ising

model was pushed out of equilibrium by a fluctuating magnetic field and its dynamics where observed. Using simulated data, some of the systems phase space was explored and mapped on the basis of expectation properties and areas of that space where found to be unique and of interest. Using this newly proposed method, the regime was explored and entropic mappings were made which reflect the dynamics.

The method implemented here was adopted from technological and methodological advancements made in the field of image processing using deep learning networks. At the time of this writing Van Der Oords initial publication on the PixelCNN algorithm has accrued over 1,900 citations in under six years. Methods like generative adversarial networks, normalizing flows and variational autoencoders are progressing concurrently and managing to model subtleties within high-dimensional distributions [83] [84] [85]. Further, methods are coming out of Google which manage to outperform all of these following models in the task of convincing image generation for full color images of size 1024x1024 [86]. These models are staggeringly large and are computed using many state-of-the-art Tensor Processing Units over the course of several days. All this is worth mentioning to show that the scope of potential statistical-mechanical systems which can be studied should in no way be limited to the 2-dimensional Ising model.

Methods of this kind and others like it, utilizing the power of NNs, should be seriously considered for exploration. The results of this work demonstrate the power of this and similar methods while the work being done outside the walls of the University prove that the surface of potential scientific exploration has only just been scratched.

Acknowledgments

I would like to start by thanking Guy Cohen for his patience and tutelage. He provided me with guidance the whole way through this process, building much of my understanding of the field from the ground up. He taught me how to approach problems and gave me the tools to solve them. I consider meeting him one of the greatest blessings of my life and certainly my career.

In addition to helping me personally, Guy also put together a fantastic group of young scientist who I can proudly say are more than just colleagues – hopefully life long friends.

They helped me with research, to excel at course work and to fully enjoy life here in the city. Thanks team.

The Tel Aviv University school of Chemistry is superb in the quality of its faculty members, the courses they offer and the environment of camaraderie and support which they foster.

I would also like to thank my family for their constant support, both in the highs and lows which came to pass through the length of this degree.

---

* gcohen@tau.ac.il

[1] W. T. Kelvin, J. Larmor, and J. P. Joule, *Mathematical and Physical Papers* (Cambridge, University Press).

[2] Ã. Brunet, T. Hocquet, and X. Leyronas, , 95.

[3] P. d. C. A. du texte Spindler, G. .-. A. du texte Meyer, and J. H. A. du texte Meerburg, "Annalen der Physik,".

[4] R. Clausius, **2**, 1.

[5] "Google Books Ngram Viewer," ().

[6] K. D. Bailey, *Social Entropy Theory*.

[7] S. Kullback and R. A. Leibler, **22**, 79.

[8] A. Jakimowicz, **22**, 452.

[9] C. Bourke, J. M. Hitchcock, and N. V. Vinodchandran, **349**, 392.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing* (Cambridge University Press) 1aAOdzK3FegC.

[11] B. J. Alder and T. E. Wainwright, **31**, 459.

[12] J. Lee, **71**, 211.

[13] K. Binder and D. Stauffer, in *Applications of the Monte Carlo Method in Statistical Physics*, Topics in Current Physics, edited by K. Binder (Springer) pp. 1–36.

[14] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press).

[15] J. M. Rickman and D. J. Srolovitz, **99**, 7993.

[16] P. M. C. de Oliveira, T. J. P. Penna, and H. J. Herrmann, "Broad Histogram Method,"

arXiv:cond-mat/9610041.

[17] A. M. Ferrenberg and R. H. Swendsen, , 5.

[18] J.-S. Wang, **127**, 10.1016/S0010-4655(00)00016-3.

[19] F. Wang and D. Landau, **86**, 2050.

[20] M. Souaille and B. Roux, **135**, 40.

[21] C. Zhou and R. N. Bhatt, **72**, 025701.

[22] E. T. Jaynes, **106**, 620.

[23] D. Frenkel and A. J. C. Ladd, **81**, 3188.

[24] C. P. Herrero and R. Ramirez, **568–569**, 70, arXiv:1307.3950.

[25] C. Peter, C. Oostenbrink, A. van Dorp, and W. F. van Gunsteren, **120**, 2652.

[26] T. S. Komatsu, N. Nakagawa, S.-i. Sasa, and H. Tasaki, **159**, 1237, arXiv:1405.0697.

[27] L. Ferreira Calazans and R. Dickman, **99**, 032137.

[28] M. Kastner and M. Promberger, , cond.

[29] G. Darbellay and I. Vajda, **45**, 1315.

[30] S.-k. Ma, , 20.

[31] R. Avinery, M. Kornreich, and R. Beck, **123**, 178102, arXiv:1709.10164.

[32] A. L. Samuel, **3**, 210.

[33] F. Rosenblatt, **65**, 386.

[34] "Science: The Goof Button - TIME," ().

[35] N. Nilsson, *Learning Machines*, Learning Machines (New York).

[36] D. Nitzan, A. Brain, and R. Duda, **65**, 206.

[37] J. Allen, , 249.

[38] P. J. Werbos, .

[39] H. Robbins and S. Monro, **22**, 400.

[40] A. N. Kolmogorov, **25**, 369, 25049284.

[41] P. S. Laplace, *ThÃ©orie analytique des probabilitÃ©s;* (Paris, Ve. Courcier).

[42] T. M. Cover and J. A. Thomas, .

[43] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, **21**, 1087.

[44] W. K. Hastings, **57**, 97.

[45] S.-K. Au and J. L. Beck, **16**, 263.

[46] "6. Phase Transitions: Introduction to Statistical Mechanics," ().

[47] P. Curie, *PropriÃ©tÃ©s magnÃ©tiques des corps Ã diverses tempÃ©ratures* (Gauthier-Villars et fils) QhMywOm$_y$*NsC.R. Fitzpatrick*, *"TheIsingmodel,"* .

[48] L. Onsager, **65**, 117.

[49] L. Onsager, **65**, 117.

[50] N. D. H. Dass, B. E. Hanlon, and T. Yukawa, **368**, 55, arXiv:hep-th/9505076.

[51] A. E. Ferdinand and M. E. Fisher, **185**, 832.

[52] F. Zwicky, **43**, 270.

[53] S. G. BRUSH, **39**, 883.

[54] T. T. Wu, B. M. McCoy, C. A. Tracy, and E. Barouch, **13**, 316.

[55] B. Nickel, **32**, 3889.

[56] S. Boukraa, A. J. Guttmann, S. Hassani, I. Jensen, J.-M. Maillard, B. Nickel, and N. Zenine, **41**, 455202.

[57] T. D. Lee and C. N. Yang, **87**, 410.

[58] K. Kawasaki, **150**, 285.

[59] U. Wolff, **62**, 361.

[60] R. J. Glauber, **4**, 294.

[61] H. Hotelling, **24**, 417.

[62] Drleft, "English: Comparison of a histogram and a kernel density estimate." (28 September 2010, 12:52 (UTC)).

[63] C. E. Shannon, , 55.

[64] J. Ziv and A. Lempel, **24**, 530 ().

[65] J. Ziv and A. Lempel, **23**, 337 ().

[66] I. M. Pu, *Fundamental Data Compression* (Butterworth-Heinemann) Nyt0HgC81I4C.

[67] D. Benedetto, E. Caglioti, and V. Loreto, **88**, 048702.

[68] J. Jiao, H. H. Permuter, L. Zhao, Y.-H. Kim, and T. Weissman, **59**, 6220, arXiv:1201.2334.

[69] G. U. Yule, **84**, 497, 2341101.

[70] A. Bulinski and A. Kozhevin, "Statistical Estimation of Conditional Shannon Entropy," arXiv:1804.08741 [math, stat].

[71] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm, "MINE: Mutual Information Neural Estimation," arXiv:1801.04062 [cs, stat].

[72] A. Nir, E. Sela, R. Beck, and Y. Bar-Sinai, **117**, 30234, 33214150.

[73] K. Sricharan, D. Wei, and A. O. Hero III, "Ensemble estimators for multivariate entropy estima-

tion," arXiv:1203.5829 [math, stat].

[74] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," (), arXiv:1601.06759 [cs].

[75] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional Image Generation with PixelCNN Decoders," (), arXiv:1606.05328 [cs].

[76] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked Autoencoder for Distribution Estimation," arXiv:1502.03509 [cs, stat].

[77] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, .

[78] J. Bergstra, R. Bardenet, Y. Bengio, and B. KÃ©gl, in *Advances in Neural Information Processing Systems*, Vol. 24 (Curran Associates, Inc.).

[79] R. Sheikhpour, M. A. Sarram, and R. Sheikhpour, **40**, 113.

[80] S. Saremi, A. Mehrjou, B. SchÃ¶lkopf, and A. HyvÃ€rinen, "Deep Energy Estimator Networks," arXiv:1805.08306 [cs, stat].

[81] P. D. Beale, **76**, 78.

[82] I. O. Morales, E. Landa, C. C. Angeles, J. C. Toledo, A. L. Rivera, J. M. Temis, and A. Frank, **10**, e0130751.

[83] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing Flows for Probabilistic Modeling and Inference," arXiv:1912.02762 [cs, stat].

[84] A. Vahdat and J. Kautz, "NVAE: A Deep Hierarchical Variational Autoencoder," arXiv:2007.03898 [cs, stat].

[85] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," arXiv:1710.10196 [cs, stat].

[86] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, , 1.